(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 546 072**

(51) Int. CI.:

G06F 12/08 (2006.01)
G06F 12/14 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **14.09.2012 E 12184447 (6)**

(97) Fecha y número de publicación de la concesión europea: **27.05.2015 EP 2709017**

(54) Título: **Dispositivo para controlar el acceso a una estructura de memoria caché**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.09.2015

(73) Titular/es:

BARCELONA SUPERCOMPUTING CENTER-CENTRO NACIONAL DE SUPERCOMPUTACIÓN (100.0%)
C/ Jordi Girona 31
08034 Barcelona, ES

(72) Inventor/es:

ABELLA FERRER, JAIME;
QUIÑONES MORENO, EDUARDO y
CAZORLA ALMEIDA, FRANCISCO JAVIER

(74) Agente/Representante:

ZEA CHECA, Bernabé

DESCRIPCIÓN

Dispositivo para controlar el acceso a una estructura de memoria caché

CAMPO DE LA INVENCIÓN

La presente invención se refiere a un procedimiento para controlar el acceso a una estructura de memoria caché que comprende varios conjuntos caché durante la ejecución de al menos un programa informático. Más específicamente, la invención se refiere a un procedimiento capaz de garantizar que a cada dirección accedida por un programa informático se le puede asociar una probabilidad real de ser asignada a cualquier conjunto caché particular de una estructura de memoria caché.

Además, la invención se refiere también a un dispositivo y un producto de programa informático para controlar el acceso a una estructura de memoria caché adecuados para realizar dicho procedimiento.

La invención se puede aplicar a sistemas en tiempo real, por ejemplo sistemas en tiempo real críticos para la seguridad tales como los sistemas de control de vuelo.

ANTECEDENTES DE LA TÉCNICA

Las memorias caché son generalmente búferes pequeños de almacenamiento rápido que se pueden emplear para almacenar información, tal como código o datos, con el fin de que un programa que se ejecuta en un dispositivo de procesamiento se ejecute más rápido. Típicamente, es más rápido para el dispositivo de procesamiento leer la memoria caché que leer una memoria principal. También, con el rápido aumento de los requisitos de procesamiento intensivo, su importancia en un sistema informático sólo irá en aumento.

Una estructura de memoria caché es conceptualmente una matriz de S*W líneas caché (conceptualmente celdas) dispuestas en S conjuntos (conceptualmente filas) y W vías (conceptualmente columnas). El conjunto (es decir, la fila) en el que se coloca un fragmento de datos en la memoria caché es determinado por la política de colocación. La política de colocación implementa una función hash que utiliza ciertos bits de la dirección de memoria en la que se almacena el dato para asignar ese fragmento de datos a un conjunto caché específico (fila). Desde el punto de vista de la memoria caché, se pueden agrupar diferentes fragmentos de datos en líneas caché, o simplemente líneas. Dado que diferentes líneas de memoria pueden colisionar en el mismo conjunto caché, los conjuntos caché consisten en un determinado número de líneas caché denominadas vías (es decir, columnas). Todos los conjuntos tienen el mismo número de vías, que determinan la asociatividad de la memoria caché. Por lo tanto, una memoria caché asociativa por conjuntos de W vías comprende W vías por conjunto. Para un conjunto dado, la vía (línea caché) en la que se coloca una línea de memoria es determinada por la política de reemplazo. En el caso de que un acceso a la memoria caché produzca como resultado un fallo, la política de reemplazo selecciona, de entre todas las vías (columnas) de un conjunto dado (fila), qué línea de memoria es desalojada para dar cabida a la nueva línea de memoria. La nueva línea de memoria es almacenada, entonces, en la línea caché cuyo contenido (una línea de memoria) acaba de ser desalojado. Cada línea puede estar compuesta por una o varias palabras. La granularidad de una palabra se mide generalmente en bytes (por ejemplo, 1, 2 o 4 bytes).

El comportamiento de temporización (*timing behaviour*) de una memoria caché está determinado principalmente por sus políticas de colocación y reemplazo. El tamaño de la línea, el tamaño de la palabra y otros parámetros de la memoria caché también pueden afectar el comportamiento de temporización de la memoria caché. Para una determinada configuración de memoria caché, tanto el tamaño de la línea como el tamaño de la palabra son fijos.

Se han propuesto memorias caché aleatorias en procesadores de alto rendimiento para eliminar conflictos de memoria caché mediante el uso de funciones hash pseudo-aleatorias *[A. Gonzalez et al. Eliminating cache conflict misses through XOR-based placement functions. In ICS, 1997.][A. Seznec and F. Bodin. Skewed-associative caches. In PARLE. 1993.][Nigel Topham and Antonio González. Randomized cache placement for eliminating conflicts. IEEE Trans. Comput., 48, February 1999]*. Sin embargo, el comportamiento de todos esos diseños de memoria caché es totalmente determinista. Esto significa que, cada vez que un determinado conjunto de datos de input para un programa hace que el programa genere un patrón de acceso patológico, este patrón se repetirá

sistemáticamente para dicho conjunto de input en todas las ejecuciones del programa. Por lo tanto, aunque se reduce la frecuencia de casos patológicos, todavía pueden aparecer sistemáticamente porque no hay manera de demostrar que su probabilidad está vinculada.

En el dominio de tiempo real el Análisis Probabilístico de Temporización (*Probabilistic Timing Analysis - PTA*) (véase, por ejemplo, *[F. J. Cazorla et al. Proartis: Probabilistically analysable real-time systems. Technical Report 7869 (http://hal.inria.fr/hal-00663329), INRIA, 2012], [D. Griffin and A. Burns. Realism in Statistical Analysis of Worst Case Execution Times. In the 10th International Workshop on Worst-Case Execution Time Analysis (WCET 2011), páginas 44-53, 2010] or [J. Hansen, S. Hissam, and G. A. Moreno. Statistical-based WCET estimation and validation. In the 9th International Workshop on Worst-Case Execution Time (WCET) Analysis, 2009]*) se ha convertido en una prometedora solución efectiva a los problemas de las técnicas actuales de análisis del tiempo de ejecución en el peor caso (*Worst-Case Execution Time – WCET*), es decir, análisis estático de temporización y análisis de temporización basado en mediciones.

El Análisis Probabilístico de Temporización impone nuevos requisitos en los diseños del hardware. Más específicamente, en el diseño de la memoria caché las técnicas de Análisis Probabilístico de Temporización requieren que el comportamiento de la temporización de los accesos a memoria pueda estar definido por el par de vectores:

$$\{\vec{l}_i, \vec{p}_i\} = \{\{l_i^1, l_i^2, ..., l_i^N\}, \{p_i^1, p_i^2, ..., p_i^N\}\}$$

en los que l_i enumera todas las posibles latencias que la jerarquía de memoria puede tomar para servir los datos y p_i su probabilidad asociada de ocurrencia. Se observa que la probabilidad de ocurrencia de una determinada latencia es diferente a su frecuencia: mientras que la frecuencia proporciona información sobre eventos pasados, las probabilidades permiten ofrecer garantías sobre la ocurrencia de eventos futuros (véase, por ejemplo, *[F. J. Cazorla et al. Proartis: Probabilistically analysable real-time systems. Technical Report 7869 (http://hal.inria.fr/hal-00663329), INRIA, 2012]*). Por lo tanto, para el caso de un Análisis Probabilístico de la Temporización de un recurso de memoria caché se requiere que para cada acceso haya una probabilidad calculable de acierto o de fallo en la memoria caché.

En *H Vandierendonck et al.: "Randomized Caches for Power-Efficiency", IEICE Transactions on Electronics, Institute of Electronics, Tokyo, vol. E86 C, no. 10, 2003, páginas 2137-2144* se presenta una memoria caché que tiene una función de índice de conjunto aleatorizada ajustada (*tuned randomized set index function*). La función de aleatorización se configura cuando se inicia la aplicación y se basa en información de perfiles generada usando inputs representativos.

Existen políticas de reemplazo aleatorias para hacer que la selección de una línea caché (celda) dentro de un conjunto caché (fila) sea aleatoria. Sin embargo, las políticas de colocación existentes son puramente deterministas en base a la dirección accedida. Por lo tanto, que los accesos a dos direcciones diferentes compitan por el mismo conjunto caché depende únicamente de sus direcciones particulares y de la función de colocación utilizada. Por lo tanto, si dos direcciones son colocadas en el mismo conjunto caché al inicio de la ejecución, éstas colisionarán en ese conjunto caché siempre durante la ejecución del programa y en todas las ejecuciones del programa. Dado que el comportamiento es puramente determinista no se puede calcular una probabilidad real. Por lo tanto, no se satisfacen las propiedades probabilísticas requeridas por el Análisis Probabilístico de la Temporización.

En la seguridad de procesadores, las memorias caché estándar no aleatorias son vulnerables a la fuga de información crítica tal como claves criptográficas. Los ataques a memorias caché estándar se basan únicamente en la diferencia de temporización entre aciertos y fallos en la memoria caché. Rompiendo el determinismo entre accesos y si aciertan o fallan mediante el uso de memorias caché de reemplazo aleatorio o similares, hace que los aciertos y fallos se produzcan con una determinada probabilidad, lo que mejora la seguridad ya que se oculta la información a los atacantes.

En general, hasta ahora sólo las memorias caché con un conjunto caché (sin que se requiera una función de colocación) y una pluralidad de vías caché que implementan un reemplazo aleatorio son adecuadas para las técnicas de Análisis Probabilístico de Temporización y para reducir vulnerabilidades de seguridad. Por desgracia,

esas memorias caché, también conocidas como memorias caché completamente asociativas, normalmente son costosas en términos de energía y área y no escalan bien.

RESUMEN DE LA INVENCIÓN

Un objeto de la presente invención es proporcionar un dispositivo para controlar el acceso a una estructura de memoria caché, que evita las desventajas de la técnica anterior.

Para lograr lo anterior, de acuerdo con un primer aspecto, la invención proporciona un dispositivo para controlar el acceso a una estructura de memoria caché que comprende varios conjuntos caché durante la ejecución de al menos un programa informático, comprendiendo el dispositivo un módulo para generar valores semilla durante la ejecución del al menos un programa informático; un módulo de función hash paramétrica para generar un identificador de conjunto caché para acceder a la estructura de memoria caché, generándose el identificador mediante la combinación de un valor semilla generado por el módulo para generar valores semilla y unos bits predeterminados de una dirección para acceder a una memoria principal asociada a la estructura de memoria caché.

De esta manera, en una estructura de memoria caché con varios conjuntos, la provisión del valor semilla garantiza que un determinado acceso tenga probabilidades reales de ser asignado a cada uno de los conjuntos caché de la estructura de memoria caché independientemente del valor de la dirección. Es decir, la invención es capaz de garantizar que a cada dirección accedida por un programa informático, a través de un módulo procesador tal como un microprocesador, se le asocie una probabilidad real de ser asignada a cualquier conjunto caché particular de una estructura de memoria caché que comprende varios conjuntos.

Es importante destacar que el valor semilla generado se puede actualizar periódicamente o cuando se produce un evento en particular. De esta manera, el valor semilla puede ser actualizado sólo al inicio o al final de la ejecución del programa informático, aunque su actualización en un momento diferente, o con una frecuencia mayor o menor también está dentro del alcance de la invención.

En consecuencia, si se cambia el valor semilla, el identificador obtenido del conjunto también cambia para una dirección fija determinada. Para el propósito de esta invención, se puede utilizar cualquier módulo de función hash paramétrica siempre y cuando haya una probabilidad real de que una dirección pueda ser asignada a cualquier conjunto. Se puede utilizar cualquier valor semilla. Sin embargo, se logra un mayor grado de aleatoriedad si el módulo de función hash paramétrica produce grandes variaciones en su identificador de conjunto de output incluso si se producen pequeños cambios en la dirección de input y/o el valor semilla, y si la semilla es ajustada a un valor aleatorio o pseudo-aleatorio.

Además, las estructuras de memoria caché con varios conjuntos que implementan funciones de colocación son más convenientes en términos de energía. Esas estructuras de memoria caché con varios conjuntos se conocen como memorias caché de correspondencia directa (*direct-mapped caches*) si sólo tienen una línea caché por conjunto (sin necesidad de una política de reemplazo) y como memorias caché asociativas por conjuntos (*set-associative caches*) si tienen más de una línea caché por conjunto (ambas políticas de colocación y reemplazo son necesarias).

Según una forma de realización preferida de la invención, la estructura de memoria caché puede comprender 2^n conjuntos, y el identificador de conjunto caché generado puede comprender n bits.

Preferiblemente, la dirección determinada comprende una secuencia de bits, un subconjunto de los cuales puede corresponder a un desplazamiento, y los bits predeterminados del identificador de conjunto de la dirección pueden comprender la secuencia de bits sin el desplazamiento.

Según otra forma de realización, la estructura de memoria caché puede comprender al menos una línea caché por conjunto, tal como una estructura de memoria caché asociativa por conjuntos o una estructura de memoria caché de correspondencia directa.

De acuerdo con otra forma de realización de la invención, el módulo para generar valores semilla durante la ejecución de al menos un programa informático puede comprender un generador de números pseudo-aleatorios, que puede ser implementado ya sea en hardware o en software. Por ejemplo, el módulo para generar valores semilla puede ser implementado en hardware por medio de elementos electrónicos (por ejemplo, puertas lógicas).

Además, el dispositivo puede comprender un módulo de memoria para almacenar los valores semilla generados, de manera que todos los valores semilla generados se pueden almacenar en el módulo de memoria. Alternativamente, cada valor semilla generado nuevamente puede reemplazar el valor semilla generado previamente.

Según otro aspecto, la invención proporciona un procedimiento de controlar el acceso a una estructura de memoria caché que comprende varios conjuntos caché durante la ejecución de al menos un programa informático, comprendiendo el procedimiento:
- Generar valores semilla durante la ejecución de al menos un programa informático;
- Recibir una dirección para acceder a una memoria principal asociada a la estructura de memoria caché;
- Generar un identificador de conjunto caché para acceder a la estructura de memoria caché, generándose el identificador mediante la combinación de un valor semilla generado y bits predeterminados de la dirección recibida.

De acuerdo con todavía otro aspecto de la invención, se proporciona un producto de programa informático que comprende instrucciones de programa para hacer que un ordenador realice el procedimiento de controlar el acceso a una estructura de memoria caché según se ha descrito anteriormente. La invención también se refiere a dicho producto de programa informático almacenado en un medio de almacenamiento (por ejemplo, un CD-ROM, un DVD, una unidad USB, en una memoria de ordenador o en una memoria de sólo lectura) o portado en una señal portadora (por ejemplo, en una señal portadora eléctrica u óptica).

La invención también proporciona una estructura de memoria caché que comprende varios conjuntos caché y el dispositivo para controlar el acceso a la estructura de memoria caché según se ha descrito anteriormente.

Otros objetos, ventajas y características de formas de realización de la invención serán evidentes para los expertos en la técnica tras el examen de la descripción, o se pueden aprender por la puesta en práctica de la invención.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

A continuación se describirán formas de realización particulares de la presente invención por medio de ejemplos no limitantes, con referencia a los dibujos adjuntos, en los que:

La figura 1 ilustra esquemáticamente una configuración de procesador que implementa una estructura de memoria caché, que comprende un dispositivo para controlar el acceso a la estructura de memoria caché de acuerdo con la invención;
La figura 2 representa esquemáticamente un registro de dirección que contiene una solicitud a la estructura de memoria caché para una dirección específica de una memoria principal proporcionada por un módulo de procesador asociado a la estructura de memoria caché;
La figura 3 ilustra un diagrama de bloques de una forma de realización de un módulo de función hash paramétrica comprendido en el dispositivo para controlar el acceso a una estructura de memoria caché que se muestra en la figura 1;
La figura 4 ilustra un diagrama de bloques de un módulo de rotación (*rotate module*) utilizado en el módulo de función hash paramétrica que se muestra en la figura 3;
La figura 5 ilustra un diagrama de bloques de un módulo XOR utilizado en el módulo de función hash paramétrica que se muestra en la figura 3.

DESCRIPCIÓN DETALLADA DE FORMAS DE REALIZACIÓN

En la siguiente descripción de formas de realización preferidas, se exponen numerosos detalles específicos para proporcionar una comprensión completa de la presente invención. Sin embargo, los expertos en la técnica

apreciarán que la invención puede ponerse en práctica sin dichos detalles específicos. En otros casos, se han ilustrado elementos bien conocidos en forma de diagrama esquemático o de bloques a fin de no oscurecer la presente invención con detalles innecesarios.

Se observa además que, a menos que se indique lo contrario, todas las funciones descritas en el presente documento pueden realizarse en hardware o en software, o alguna combinación de los mismos. En formas de realización preferidas, sin embargo, las funciones son realizadas por un procesador, tal como un ordenador o un procesador electrónico de datos, de acuerdo con un código, tal como un código de programa informático, software, y/o circuitos integrados que están codificados para realizar dichas funciones, a menos que se indique lo contrario.

La ejecución de programas informáticos en un sistema informático (por ejemplo, un ordenador personal) se refiere al proceso por el cual el sistema informático, a través de su procesador, lleva a cabo las instrucciones del programa informático. Algunas de estas instrucciones pueden tener que acceder a datos almacenados en una dirección predeterminada de una memoria principal comprendida en el sistema informático. De esta manera, cuando el procesador intenta acceder a la memoria principal según la dirección predeterminada para obtener los datos necesarios, en primer lugar verifica si los datos requeridos están almacenados en una estructura de memoria caché asociada a la memoria principal. Esta es la razón por la cual una estructura de memoria caché comprende búferes generalmente pequeños de almacenamiento rápido que se pueden usar para almacenar información, que permiten que un procesador tenga un acceso más rápido y eficiente a la información, es decir, es más rápido para un procesador leer la memoria más pequeña de una estructura de memoria caché que leer una memoria principal.

Según se ha descrito anteriormente, una estructura de memoria caché es conceptualmente una matriz de S*W líneas caché (conceptualmente celdas) dispuestas en S conjuntos (conceptualmente filas) y W vías (conceptualmente columnas). El conjunto (es decir, la fila) en el que se coloca un fragmento de datos en la memoria caché es determinado por la *política de colocación*. La política de colocación implementa una función hash que utiliza ciertos bits de la dirección de memoria para asignar cada línea particular de memoria a un conjunto caché específico (fila). Dado que diferentes líneas de memoria pueden colisionar en el mismo conjunto caché, los conjuntos caché consisten en un determinado número de líneas denominadas vías (es decir, columnas). Todos los conjuntos tienen el mismo número de vías, que determinan la asociatividad de la memoria caché. Por lo tanto, una memoria caché asociativa por conjuntos de W vías comprende W vías por conjunto. Para un conjunto dado, la vía (línea caché) en la que se coloca una línea de memoria es determinada por la *política de reemplazo*. En el caso de que un acceso a la memoria caché produzca como resultado un fallo (*miss*), la política de reemplazo selecciona, de entre todas las vías (columnas) de un determinado conjunto (fila), qué línea de memoria es desalojada para dar cabida a la nueva línea de memoria. La nueva línea de memoria es almacenada, entonces, en la línea caché cuyo contenido (una línea de memoria) acaba de ser desalojado. Cada línea puede estar compuesta por una o varias palabras. La granularidad de una palabra se mide generalmente en bytes (por ejemplo, 1, 2 o 4 bytes).

La figura 1 ilustra una configuración de procesador en la que se implementa una estructura de memoria caché 10. La estructura de memoria caché 10 comprende una pluralidad de vías W1-W8 y una pluralidad de conjuntos S1-S4 (es decir, una matriz de 4*8 líneas caché). La estructura de memoria caché 10 está acoplada a una unidad de procesamiento central (*central processing unit - CPU*) 11 y una memoria principal 12. Además, la estructura de memoria caché 10 también comprende un módulo lógico de colocación de conjuntos 13 que determina qué conjunto de los conjuntos caché S1-S4 se utiliza para asignar una determinada dirección para acceder a la memoria principal 12, producida por la CPU 11. El módulo lógico de colocación de conjuntos 13 recibe el identificador de conjunto caché para acceder a la estructura de memoria caché 10 procedente de un dispositivo 14 para controlar el acceso a la estructura de memoria caché de acuerdo con la invención.

El dispositivo de control 14 comprende un módulo de función hash paramétrica 15 que combina un valor semilla 16 generado por, por ejemplo, un generador de números pseudo-aleatorios (no mostrado) y unos bits predeterminados 17 de la dirección determinada para acceder a la memoria principal 12.

En este punto, es importante señalar que el generador de números pseudo-aleatorios puede ser implementado ya sea en hardware o en software, aunque cualquier otra forma de obtener el valor semilla está dentro del alcance de la invención.

La invención también propone la actualización del valor semilla 16 sólo al inicio o al final de la ejecución del programa informático, aunque la actualización en un momento diferente, o con una frecuencia mayor o menor también está dentro del alcance de la invención.

En consecuencia, si se cambia el valor semilla, el identificador de conjunto obtenido cambia aunque la dirección siga siendo la misma. Se puede utilizar cualquier módulo de función hash paramétrica siempre y cuando haya una probabilidad real de que una dirección se pueda asignar a cualquier conjunto. Se puede utilizar cualquier valor semilla. Sin embargo, los resultados del Análisis Probabilístico de Temporización son mejores (tiempos de ejecución más bajos) si el módulo de función hash paramétrica produce grandes variaciones en su identificador de conjunto de output, incluso si se producen pequeños cambios en la dirección de input y/o el valor semilla, y si la semilla es ajustada a un valor aleatorio o pseudo-aleatorio.

Además, el dispositivo de control 14 también puede comprender un registro (no mostrado) para almacenar el valor semilla generado 16.

La figura 2 ilustra un ejemplo de una dirección 20 para acceder a la memoria principal 12, producida por la CPU 11, los bits predeterminados 17 que se combinan con el valor semilla 16 para obtener el identificador de conjunto caché para acceder a la estructura de memoria caché 10. Básicamente, la dirección 20 comprende tres partes de bits:
- Una etiqueta 21;
- Un índice 22;
- Un desplazamiento 23.

Como ejemplo ilustrativo, si las direcciones son de 16 bits de ancho y el tamaño de la línea caché es 64 bytes (2^6 bytes - en general, el número de bytes es 2^n) entonces, para la estructura de memoria caché que se muestra en la figura 1 (4 conjuntos S1-S4, 8 vías W1-W8), los bits de dirección se clasifican de la siguiente manera: los 8 bits de mayor peso son la etiqueta 21; los 2 bits siguientes (es decir, teniendo en cuenta que la memoria caché comprende 4 conjuntos, se requieren 2 bits para direccionar todos los conjuntos) son el índice 22; y los bits de menor peso son el desplazamiento 23. De esta manera, si la dirección es "0011 0011 0100 0111" (en binario) la etiqueta 21 es "0011 0011", el índice 22 es "01", y el 23 desplazamiento es "00 0111".

Las políticas de colocación convencionales utilizan el índice 22 ("01" en el ejemplo) para elegir el conjunto caché accedido proporcionando esos bits al módulo lógico de colocación de conjuntos 13. En el ejemplo, se accede al conjunto S1 si los bits de índice son "00"; se accede al conjunto S2 si los bits de índice son "01" (como en el ejemplo); se accede al conjunto S3 si los bits de índice son "10"; y se accede al conjunto S4 si los bits de índice son "11". Se conocen otras políticas de colocación que combinan los bits de la etiqueta 21 y del índice 22 para generar un identificador de conjunto caché. Sin embargo, estas políticas de colocación son claramente deterministas ya que el identificador de conjunto caché generado depende únicamente de la dirección accedida.

El dispositivo 14 según la invención combina los bits de la etiqueta 21 y del índice 22 (es decir, los bits predeterminados 17 de la dirección 20) y el valor semilla 16 para generar un identificador de conjunto caché mediante el módulo de función hash paramétrica 15.

La Figura 3 muestra una forma de realización de un módulo de función hash paramétrica 15. Obviamente, la combinación del valor semilla 16 y los bits predeterminados 17 de la dirección 20 (es decir, en la presente forma de realización, la etiqueta 21 y el índice 22) se puede realizar de diferentes maneras, con el propósito de generar un identificador de conjunto caché para acceder a la estructura de memoria caché, por lo que el módulo de función hash paramétrica 15 puede tener diferentes configuraciones.

En la forma de realización mostrada en la figura 3, el módulo de función hash paramétrica 15 comprende un registro 30 para recibir y almacenar los bits predeterminados 17 de la dirección 20. Debido a que en la presente forma de realización la dirección tiene 32 bits, los bits predeterminados de la dirección son los bits 5 a 31 de la dirección (es decir, la dirección 20 sin el desplazamiento 23).

Además, el módulo de función hash paramétrica 15 también comprende un registro 36 para recibir y almacenar el valor semilla 16 (en la presente forma de realización el valor semilla comprende 32 bits); una pluralidad de módulos

31a, 31b, 31c, 31d para rotar los bits predeterminados 17 de la dirección 20 en base a ya sea algunos bits 32a, 32b, 32c, 32d del valor semilla 16 o algunos bits de los bits predeterminados 17; un módulo 33 para concatenar los bits del valor semilla 16, los bits predeterminados 17 de la dirección 20, y los bits obtenidos en el output de cada módulo de rotación 31a, 31b, 31c, 31d; y una pluralidad de módulos lógicos de plegado o reducción mediante XOR (*XOR-folding logic modules*) 34a, 34b, 34c, 34d, 34e de diferentes longitudes para adaptar el número de bits del valor binario concatenado a un valor binario que tiene tantos bits como sea necesario para indexar todos los conjuntos caché. El output del módulo lógico de plegado mediante XOR 34a corresponde al identificador de conjunto caché y es recibido por un registro 35, que proporciona el identificador al módulo lógico de colocación de conjuntos 13 para acceder a la estructura de memoria caché 10 y para obtener o proporcionar datos al conjunto caché. Según se ha descrito anteriormente, el módulo lógico de colocación de conjuntos 13 determina qué conjunto de los conjuntos caché S1-S4 se utiliza para asignar la dirección indicada 20.

Después, se describe a partir de la figura 4 el funcionamiento de un módulo de rotación 31a; 31b; 31c; 31d comprendido en el módulo de función hash paramétrica 15 que se muestra en la figura 3.

Básicamente, en la presente forma de realización, cada módulo de rotación comprende un elemento lógico de rotación 41 cuyo objeto es hacer rotar N posiciones a la izquierda un valor binario almacenado en un registro de valor 40, en el que N es determinado por un registro de control 42. El output del elemento lógico de rotación 41 es recibido y almacenado en un registro 43. Por lo tanto, los N bits de más a la izquierda del valor binario del registro de valor 40 se convierten en los N bits de más a la derecha del registro de output 43 y los M bits restantes de más a la derecha del valor binario del registro de valor 40 se convierten en los M bits de más a la izquierda del registro de output 43. Aunque no es una limitación, se recomienda que el número de bits del valor binario sea una potencia de dos (por ejemplo, 2 bits, 4 bits, 8 bits, etc.) y que el registro de control 42 tenga tantos bits como el logaritmo en base 2 del tamaño del valor binario. Por ejemplo, si el valor binario es de 32 bits de ancho, el registro de control 42 debe tener 5 bits de modo que pueda tomar 32 valores diferentes que puedan producir las 32 rotaciones diferentes del valor binario (de 0 a 31 rotaciones de bits).

El siguiente ejemplo ilustra cómo funciona un módulo de rotación 31a; 31b; 31c; 31d. Si el valor binario es 00110111 (en binario) y el registro de control 42 es 101, entonces el registro de output 43 será 11100110. El registro de control 42 hace que el valor binario sea rotado 5 (101 en binario) posiciones a la izquierda de modo que los 5 bits de más a la izquierda del valor binario (00110) se convierten en los 5 bits de más a la derecha del valor binario almacenado en el registro de output 43 y los restantes bits de más a la derecha 3 del valor binario (111) se convierten en los 3 bits de más a la izquierda del valor binario almacenado en el registro de output 43.

Volviendo a la figura 3, si esta explicación es aplicada a los módulos de rotación mostrados en dicha figura, la correspondencia es la siguiente:

- Módulo de rotación 31a (que corresponde al elemento lógico de rotación 41 de la figura 4): el valor binario comprende los bits 5 a 31 (es decir, los bits predeterminados 17) de la dirección 20, que se almacenan en el registro de dirección 30 (que corresponde al registro de valor binario 40 de la figura 4); el registro de control 32a corresponde al registro de control 42 de la figura 4, almacenando el registro de control 32a los bits 5 a 9 de los bits predeterminados 17 de la dirección, que también son los bits 0 a 4 del registro de dirección 30. El output del módulo de rotación 31a es enviado al módulo de concatenación 33.

- Módulo de rotación 31b (que corresponde al elemento lógico de rotación 41): el valor binario también comprende los bits 5 a 31 (es decir, los bits predeterminados 17) de la dirección 20, que se almacenan en el registro de dirección 30 (que corresponde al registro de valor binario 40 de la figura 4); el registro de control 32b corresponde al registro de control 42 de la figura 4, almacenando el registro de control 32b los bits 10 a 14 de los bits predeterminados 17 de la dirección, que también son los bits 5 a 9 del registro de dirección 30. El output del módulo de rotación 31b es enviado al módulo de concatenación 33.

- Módulo de rotación 31c (que corresponde al elemento lógico de rotación 41): el valor binario también comprende los bits 5 a 31 (es decir, los bits predeterminados 17) de la dirección 20, que se almacenan en el registro de dirección 30 (que corresponde al registro de valor binario 40 de la figura 4); el registro de control 32c corresponde al

registro de control 42 de la figura 4, almacenando el registro de control 32c los bits 0 a 4 del valor semilla 16. El output del módulo de rotación 31c es enviado al módulo de concatenación 33.

- Módulo de rotación 31d (que corresponde al elemento lógico de rotación 41): el valor binario comprende los bits 5 a 31 (es decir, los bits predeterminados 17) de la dirección 20, que se almacenan en el registro de dirección 30 (que corresponde al registro de valor binario 40 de la figura 4); el registro de control 32d corresponde al registro de control 42 de la figura 4, almacenando el registro de control 32d los bits 5 a 9 del valor semilla 16. El output del módulo de rotación 31d es enviado al módulo de concatenación 33.

Más específicamente, uno de los inputs para todos los módulos de rotación 31a, 31b, 31c, 31d son los bits de la dirección 20 descartando los bits de desplazamiento 23. En la forma de realización particular de la figura 3, hay 5 bits de desplazamiento y la dirección 20 es de 32 bits de ancho, por lo que se utilizan 27 bits de la dirección. Éstos se amplían con 5 bits a la izquierda, cuyo valor es 00000, por lo que hay 32 bits en total. Cada módulo de rotación utiliza un conjunto diferente de bits como registro de control 32a; 32b; 32c; 32d. Estos bits se utilizan para determinar cuántas posiciones se rota el valor binario (es decir, los bits predeterminados 17 de la dirección 20 ampliados con los 5 bits por la izquierda cuyo valor es 00000). En la figura 3, por ejemplo, el módulo de rotación de más a la izquierda 31a utiliza los 5 bits de menos peso de la dirección 20 después de descartar el desplazamiento. En otras palabras, utiliza los 5 bits de más a la derecha del par etiqueta 21 e índice 22 (bits en la posición 0, 1, 2, 3, 4 contando de derecha a izquierda). El segundo módulo de rotación de más a la izquierda 31b utiliza los siguientes 5 bits de menos peso del par etiqueta 21 e índice 22 (bits en las posiciones 5 a 9). Los otros dos módulos de rotación 31c, 31d usan bits del valor semilla 16. En particular, usan los bits 0 a 4 y los bits 5 a 9, respectivamente.

De esta manera, el módulo de concatenación 33 concatena los bits 5 a 31 de la dirección 20, los bits 0 a 31 del valor semilla 16, el output del primer módulo de rotación 31a (es decir, 32 bits, que son el resultado de hacer rotar el valor binario que comprende los bits predeterminados 17 con 5 bits por la izquierda cuyo valor es 00000), el output del segundo módulo de rotación 31b (es decir, 32 bits) el output del tercer módulo de rotación 31c (es decir, 32 bits), y el output del cuarto módulo de rotación 31d (es decir, 32 bits), con lo que se obtiene un único valor binario de 187 bits (27 + 32 + 32 + 32 + 32 + 32 = 187 bits).

A continuación, varios módulos XOR 34a, 34b, 34c, 34d, 34e se aplican en cascada al output del módulo de concatenación 33 hasta que el valor binario tiene tantos bits como sea necesario para indexar todos los conjuntos caché. En consecuencia, los módulos XOR permiten reducir el número de bits del valor binario hasta que el valor binario tiene tantos bits como sea necesario para indexar todos los conjuntos caché.

El funcionamiento de un módulo XOR se describe de acuerdo con la figura 5. Como puede verse en la figura, el módulo XOR está configurado para el caso en que el valor binario tiene 9 bits, aunque su funcionamiento es el mismo independientemente del número de bits del valor binario utilizado como input. Básicamente, el módulo XOR divide en dos mitades los bits del valor binario almacenado en un primer registro 50 y lleva a cabo una operación XOR 51a, 51b, 51c, 51d entre cada par de bits en posiciones simétricas en cada una de las mitades, es decir, XOR entre los bits 0 y 7, los bits 1 y 6, los bits 2 y 5, y los bits 3 y 4. Si el número de bits es impar, entonces el bit de mayor peso simplemente es propagado al output (este es el caso que se muestra en la figura 5). La operación XOR con 2 bits de input produce un "1" si los bits de input son diferentes (es decir, uno de ellos es "0" y el otro es "1") y un "0" si coinciden (ambos son o bien "0" o "1"). El valor binario obtenido se almacena en el registro de output 52.

Por ejemplo, si el valor binario de la concatenación proporcionado por el módulo de concatenación 33 tiene 200 bits, y el número de conjuntos caché es 32, se debe hacer XOR de los 200 bits hasta tener sólo 5 bits para indexar todos los conjuntos caché (5 es el logaritmo en base 2 de 32). Por lo tanto, diferentes niveles de módulos XOR reducirán el número de bits de 200 a 100, de 100 a 50, de 50 a 25, de 25 a 13, de 13 a 7, y de 7 a 5.

Obsérvese que siempre que el número de bits del input es menor que dos veces el número de bits requerido para el índice de conjuntos (por ejemplo, 7 es menor que dos veces 5), la lógica XOR se aplica sólo para tantos pares de bits como el número de bits de input (7 en el ejemplo) menos el número de bits de output requeridos (5 en el ejemplo), lo que significa que se aplica sobre 2 pares de bits. Para ello, simplemente se propagan tantos bits como sea necesario desde el lado izquierdo del input (3 bits en este caso) y los bits restantes (4 bits en el ejemplo) son

plegados o reducidos mediante XOR (*XOR-folded*) de modo que se obtiene como output esos bits propagados (3 bits) y el resultado de hacer XOR de los pares (2 bits).

Volviendo a la figura 3, y teniendo en cuenta la descripción realizada para la figura 5, el primer módulo XOR 34e recibe un valor binario de 187 bits (según se ha descrito anteriormente) procedente del módulo de concatenación 33 y lo transforma en un valor binario de 94 bits. El segundo módulo XOR 34d recibe y transforma dicho valor binario de 94 bits en un valor binario de 47 bits. Dicho valor de output es recibido por el tercer módulo XOR 34c, que transforma el valor binario de 47 bits en un valor binario de 24 bits. Dicho valor binario de 24 bits es recibido por el cuarto módulo XOR 34b, que lo transforma en un valor binario de 12 bits. El valor binario de 12 bits es enviado al quinto módulo XOR 34a que lo transforma en un valor que tiene tantos bits como sea necesario para indexar todos los conjuntos caché de la estructura de memoria caché. Una vez se ha obtenido el valor apropiado, es almacenado en el registro de output 35, que proporciona el valor binario al módulo lógico de colocación de conjuntos 13 para acceder a la estructura de memoria caché 10.

Aunque esta invención se ha descrito en el contexto de ciertas formas de realización y ejemplos preferentes, los expertos en la técnica entenderán que la presente invención se extiende más allá de las formas de realización divulgadas específicamente a otras formas de realización y/o usos alternativos de la invención y modificaciones obvias y equivalentes de las mismas. Por lo tanto, se pretende que el alcance de la presente invención descrita en este documento no debe estar limitado por las formas de realización particulares descritas anteriormente, sino que se debería determinar sólo por una lectura imparcial de las reivindicaciones que siguen.

Hay que señalar que se recomienda añadir bits a la dirección de input sin bits de desplazamiento, pero no es obligatorio. Téngase en cuenta que el número de bits utilizados para el registro de control 42 del módulo de rotación 41 está configurado para que sea el logaritmo en base 2 del tamaño del registro de valor binario 40, pero se podría utilizar otro número de bits.

Téngase en cuenta que se podría utilizar cualquier combinación de dirección y bits semilla en cualquiera de los inputs de los módulos de rotación.

La invención no está limitada a ninguna de esas decisiones y se podría utilizar cualquier otro módulo de función hash paramétrica 15 que utilice módulos de rotación, módulos XOR o cualquier otro tipo de lógica que caiga dentro del alcance de la invención, siempre y cuando los inputs del módulo de función hash paramétrica incluyan la dirección y una semilla.

Además, aunque las formas de realización de la invención descritas con referencia a los dibujos comprenden aparatos informáticos y procesos realizados en aparatos informáticos, la invención se extiende también a programas informáticos, particularmente programas informáticos sobre o en un portador, adaptados para poner en práctica la invención. El programa puede estar en forma de código fuente, código objeto, un código intermedio entre código fuente y objeto tal como en una forma compilada parcialmente, o en cualquier otra forma adecuada para su uso en la implementación de los procedimientos según la invención. El portador puede ser cualquier entidad o dispositivo capaz de portar el programa.

Por ejemplo, el portador puede comprender un medio de almacenamiento, tal como una ROM, por ejemplo un CD ROM o una ROM semiconductora, o un medio de grabación magnético, por ejemplo un disquete o disco duro. Además, el portador puede ser un portador transmisible tal como una señal eléctrica u óptica, que puede ser transmitido a través de cable eléctrico u óptico o por radio o por otros medios.

Cuando el programa está incorporado en una señal que puede ser transmitida directamente por un cable u otro dispositivo o medio, el portador puede estar constituido por dicho cable u otro dispositivo o medio.

Alternativamente, el portador puede ser un circuito integrado en el que está integrado el programa, estando adaptado el circuito integrado para realizar, o para uso en la realización de, los procesos relevantes.

REIVINDICACIONES

1. Un dispositivo para controlar el acceso a una estructura de memoria caché que comprende varios conjuntos caché durante la ejecución de al menos un programa informático, comprendiendo el dispositivo:
- un módulo para generar valores semilla aleatorios o pseudo-aleatorios durante la ejecución del al menos un programa informático;
- un módulo de función hash paramétrica para generar un identificador de conjunto caché para acceder a la estructura de memoria caché, generándose el identificador mediante la combinación de un valor semilla generado por el módulo para generar valores semilla y unos bits predeterminados de una dirección para acceder a una memoria principal asociada a la estructura de memoria caché.

2. El dispositivo según la reivindicación 1, en el que la estructura de memoria caché comprende 2^n conjuntos, y en el que el identificador de conjunto caché generado comprende n bits.

3. El dispositivo según cualquiera de las reivindicaciones 1 o 2, en el que la dirección determinada comprende una secuencia de bits, una parte de los cuales corresponde a un desplazamiento, y en el que los bits predeterminados de la dirección comprenden la secuencia de bits sin el desplazamiento.

4. El dispositivo según cualquiera de las reivindicaciones 1 a 3, en el que la estructura de memoria caché comprende al menos una línea caché por conjunto.

5. El dispositivo según cualquiera de las reivindicaciones 1 a 4, en el que el módulo para generar valores semilla comprende un generador de números pseudo-aleatorios.

6. El dispositivo según cualquiera de las reivindicaciones 1 a 5, que comprende además un módulo de memoria para almacenar los valores semilla generados.

7. Un procedimiento de controlar el acceso a una estructura de memoria caché que comprende varios conjuntos caché durante la ejecución de al menos un programa informático, comprendiendo el procedimiento:
- generar unos valores semilla aleatorios o pseudo-aleatorios durante la ejecución del al menos un programa informático;
- recibir una dirección para acceder a una memoria principal asociada a la estructura de memoria caché;
- generar un identificador de conjunto caché para acceder a la estructura de memoria caché, generándose el identificador mediante la combinación de un valor semilla generado y unos bits predeterminados de la dirección recibida.

8. Producto de programa informático que comprende instrucciones de programa para hacer que un ordenador realice un procedimiento de controlar el acceso a una estructura de memoria caché que comprende varios conjuntos caché según la reivindicación 7.

9. Producto de programa informático según la reivindicación 8, almacenado en un medio de almacenamiento.

10. Producto de programa informático según la reivindicación 8, portado en una señal portadora.

11. Una estructura de memoria caché que comprende varios conjuntos caché y el dispositivo para controlar el acceso a la estructura de memoria caché según cualquiera de las reivindicaciones 1 a 6.

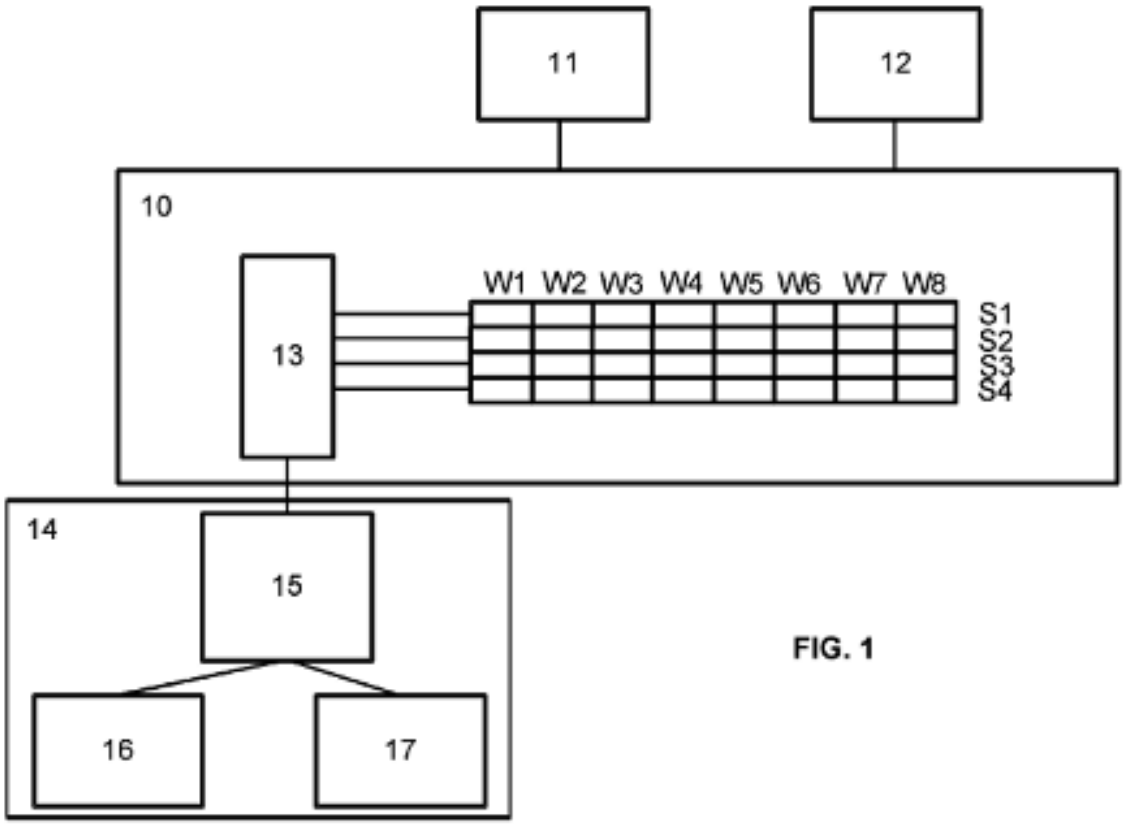


FIG. 1

FIG. 2

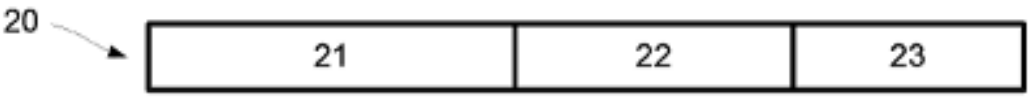
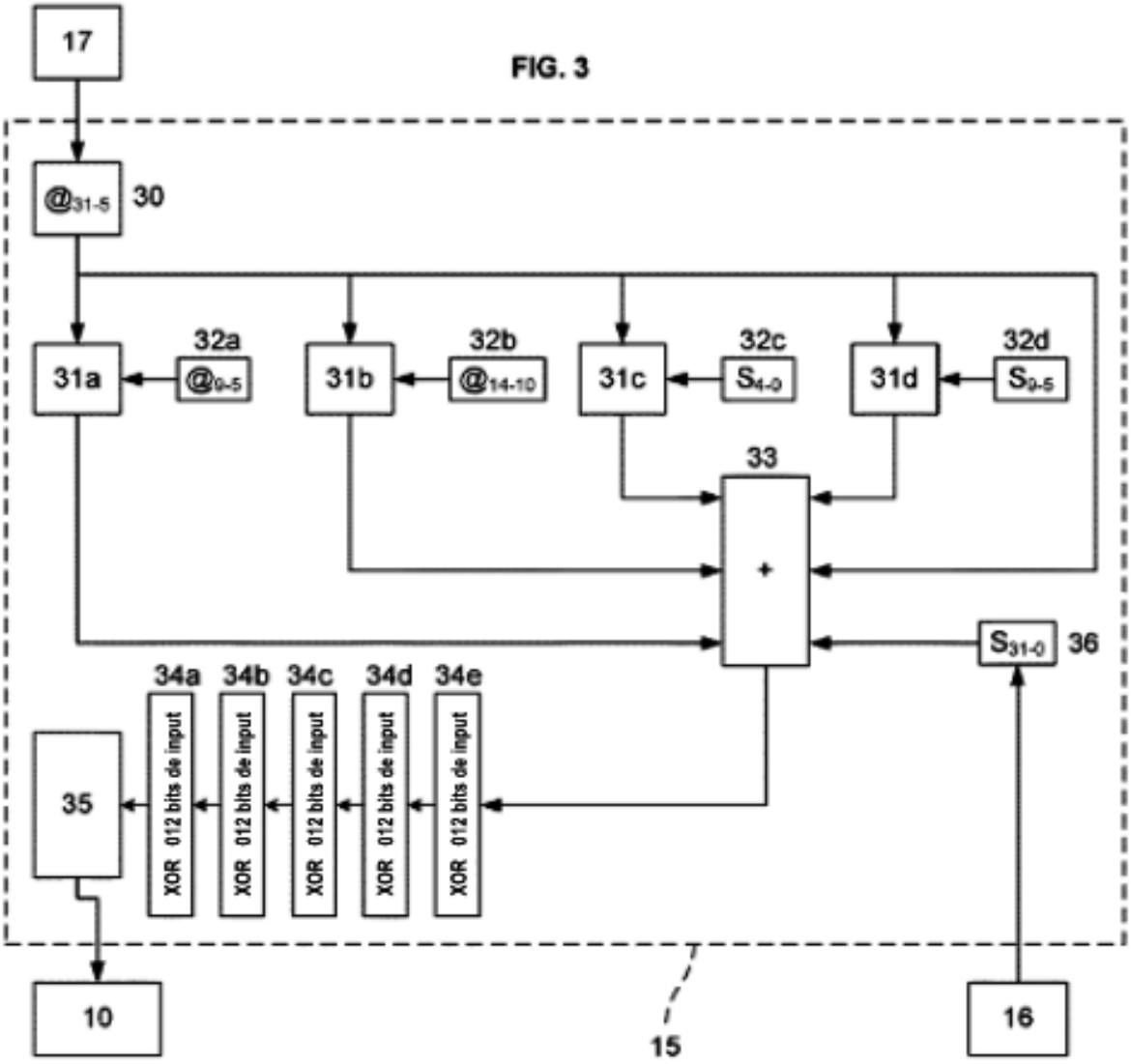


FIG. 3


17
@31-5
30
31a
32a
@9-5
31b
32b
@14-10
31c
32c
S4-0
31d
32d
S9-5
33
+
S31-0
36
34a
34b
34c
34d
34e
XOR 012 bits de input
XOR 012 bits de input
XOR 012 bits de input
XOR 012 bits de input
XOR 012 bits de input
35
10
15
16

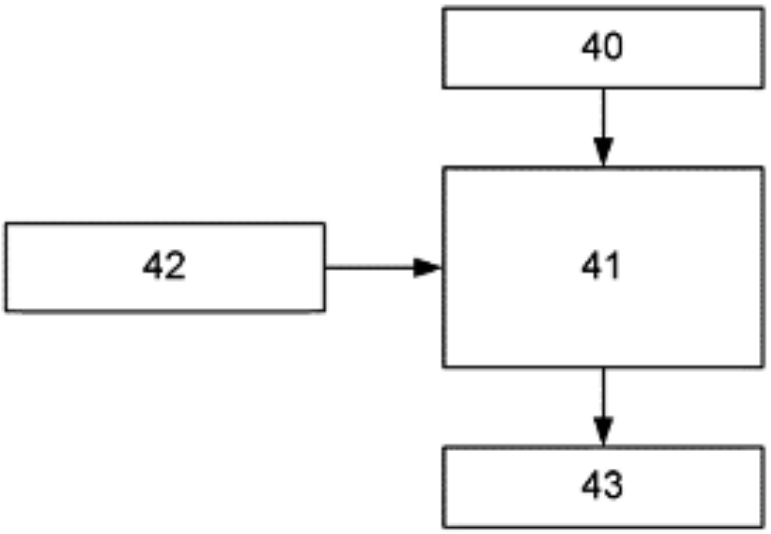


FIG. 4

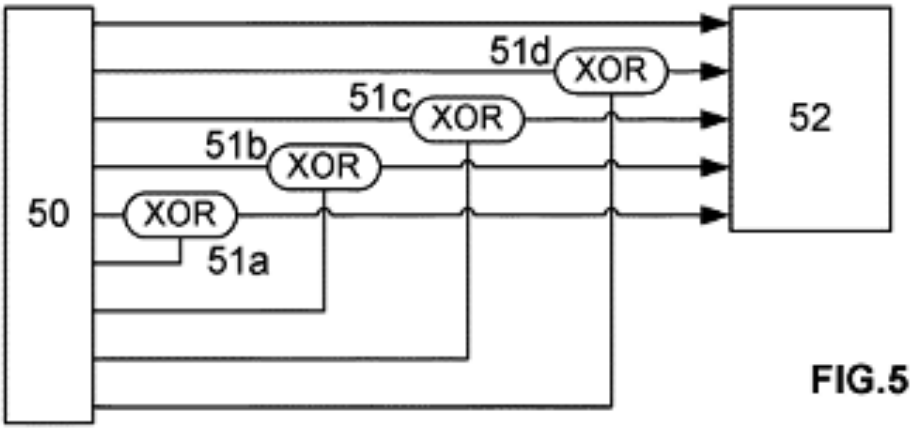


FIG.5